



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER OF PATENTS AND TRADEMARKS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/954,522	09/11/2001	David Hitz	103.1002.12	8740

22883 7590 05/20/2003  
SWERNOFSKY LAW GROUP PC  
P.O. BOX 390013  
MOUNTAIN VIEW, CA 94039-0013

EXAMINER

WASSUM, LUKE S

ART UNIT PAPER NUMBER

2177

DATE MAILED: 05/20/2003

17

Please find below and/or attached an Office communication concerning this application or proceeding.

SK

# Office Action Summary

Application No.

09/954,522

Applicant(s)

HITZ ET AL.

Examiner

Luke S. Wassum

Art Unit

2177

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

## Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

## Status

- 1) ☒ Responsive to communication(s) filed on 22 January, 13 February and 12 May 2003.
- 2a) ☐ This action is FINAL. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

## Disposition of Claims

- 4) ☒ Claim(s) 3-7, 10-17, 20-27 and 30-33 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 3-5, 7, 10-15, 17, 20-25, 27 and 30-33 is/are rejected.
- 7) ☒ Claim(s) 6, 16 and 26 is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

## Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 11 September 2001 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
- 11) ☐ The proposed drawing correction filed on \_\_\_\_\_ is: a) ☐ approved b) ☐ disapproved by the Examiner.  
If approved, corrected drawings are required in reply to this Office action.
- 12) ☐ The oath or declaration is objected to by the Examiner.

## Priority under 35 U.S.C. §§ 119 and 120

- 13) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).  
a) ☐ All b) ☐ Some \* c) ☐ None of:  
1. ☐ Certified copies of the priority documents have been received.  
2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.  
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).  
\* See the attached detailed Office action for a list of the certified copies not received.
- 14) ☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).  
a) ☐ The translation of the foreign language provisional application has been received.
- 15) ☒ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

## Attachment(s)

- 1) ☒ Notice of References Cited (PTO-892) 4) ☐ Interview Summary (PTO-413) Paper No(s). \_\_\_\_\_
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948) 5) ☐ Notice of Informal Patent Application (PTO-152)
- 3) ☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) \_\_\_\_\_ 6) ☐ Other:

## DETAILED ACTION

### *Continued Examination Under 37 CFR 1.114*

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 13 February 2003 has been entered.

### *Priority*

2. This application claims priority under U.S.C. § 120, as a continuation of application serial number 09/153,094, now U.S. Patent 6,289,356, filed 14 September 1998, which is a continuation of application serial number 09/108,022, now U.S. Patent 5,963,962, filed 30 June 1998, which is a continuation of application serial number 08/454,921, now U.S. Patent 5,819,292, filed 31 May 1995, which is a continuation of application serial number 08/071,643, now abandoned, filed 3 June 1993.

### *Response to Amendment*

3. The Applicants' amendment, filed 22 January 2003, has been received, entered into the record, and considered.

4. Furthermore, the Applicants' supplemental amendment, filed 12 May 2003, has been received, entered into the record, and considered.

Art Unit: 2177

5. As a result of the amendments, claims 3, 5, 7, 13, 15, 17, 23-25, 27 and 30-33 have been amended, and claims 8, 9, 18, 19, 28 and 29 have been canceled. Claims 1 and 2 have been previously canceled. Claims 3-7, 10-17, 20-27 and 30-33 remain pending in the application.

### *The Invention*

6. The present invention is directed to a method and system for maintaining a file system wherein membership in multiple concurrently existing file systems is possible for each block.

### *Specification*

7. As a result of the amendments to the specification and title of the application, the examiner withdraws all pending objections to the specification.

### *Claim Objections*

8. As a result of the amendments to the claims, the examiner withdraws all pending objections to the claims.

9. Claims 10, 20 and 30 are objected to because of the following informalities:

The claims cite the limitation that the system copies the root node. The limitation should cite copying the root *inode*.

10. Claim 33 is objected to because of the following informalities:

The claim is for a system, but includes 'comprising the steps of:' language, which is proper for a method claim. A system claim should use 'comprising:' language.

Art Unit: 2177

Appropriate correction is required.

*Claim Rejections - 35 USC § 112*

11. The Applicants have responded to a number of claim rejections under 35 U.S.C § 112.
12. Regarding the definition of "special files", the examiner interprets the Applicants' comments as meaning that the term "special files" means the blkmap file and the inode file, and no others.
13. Regarding the definition of "regular files", the examiner interprets the Applicants' comments as meaning that the term "regular files" means all other files of the operating system other than the blkmap file and the inode file.
14. Given these interpretations, and also in view of the Applicants' remarks and claim amendments, the examiner withdraws the pending claim rejections based on 35 U.S.C. § 112.
15. The following is a quotation of the second paragraph of 35 U.S.C. 112:  

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.
16. Claims 5, 6, 15, 16, 25 and 26 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Art Unit: 2177

17. Independent claims 5, 15 and 25 recite limitations "queuing dirty inodes..." and "requeuing any of said dirty inodes...". However, there is no claimed de-queuing step, which renders the 'requeuing' step indefinite, since the claimed dirty inodes should still be queued, and so not need to be requeued.

18. Claims 6, 16 and 26, incorporating the limitations of their respective parent claims, are also rendered indefinite.

*Claim Rejections - 35 USC § 103*

19. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

20. This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

Art Unit: 2177

21. Claims 3, 4, 13, 14, 23 and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Bach** ("The Design of the UNIX® Operating System") in view of **Chutani et al.** ("The Episode File System") in view of **Nishigaki et al.** (U.S. Patent 5,043,871).

22. Regarding claims 3 and 13, **Bach** teaches a memory and method of maintaining data in a storage system substantially as claimed, comprising the internal representation of files in the UNIX operating system, including the maintenance of inodes and blocks (see chapter 4, pages 60-90).

**Bach** does not explicitly teach a memory and method of maintaining data wherein usage bits are maintained indicating membership in an active file system and a read-only file system.

**Chutani et al.**, however, teaches a memory and method of maintaining data wherein usage bits are maintained indicating membership in an active file system and a read-only file system (see page 44, fourth full paragraph; see also page 46, seventh paragraph; see also Figures 1 and 2; see also the disclosure that an anode is analogous to an inode in a UNIX system, page 45, first full paragraph).

It would have been obvious to one of ordinary skill in the art at the time of the invention to maintain a record of changes in the storage system, i.e. a fileset clone, since such a feature would be very important to the administrative operators' implementation, since the administrative tools could operate on the clones instead of on the read-write data for their work, greatly reducing the amount of time they require exclusive access to the read-write data (see page 44, fourth full paragraph).

Neither **Bach** nor **Chutani et al.** explicitly teaches a memory and method of maintaining data wherein the storage system is capable of storing multiple usage bits for multiple read-only copies of a file system.

**Nishigaki et al.**, however, teaches a memory and method of maintaining data wherein the storage system is capable of storing multiple read-only copies of a file system (see disclosure of multiple backup version page tables and multiple backup version slot maps, analogous to the claimed multiple usage bits and multiple read-only copies of a file system, col. 12, lines 16-46; see also Figure 8).

It would have been obvious to one of ordinary skill in the art at the time of the invention to incorporate the teachings of **Nishigaki et al.** with those of **Bach** and **Chutani et al.**, since all three are concerned with the same field of endeavor, that is, managing a file system.

Furthermore, it would have been obvious to one of ordinary skill in the art at the time of the invention to allow the storage of multiple copies of read-only copies of a file system, since this would allow the storage of multiple backup versions (see col. 12, lines 28-32), providing the advantage of allowing the recovery of backups older than merely the last one.

23. Regarding claim 23, **Bach** teaches a system for maintaining data in a storage system substantially as claimed, comprising a processor (inherent in a computer system), a storage system (inherent in a file system), and a memory storing information including instructions, the instructions executable by the processor to record a plurality of data about a plurality of blocks of data in the storage system, the instructions comprising the steps of maintaining the internal representation of

Art Unit: 2177

files in the UNIX operating system, including the maintenance of inodes and blocks (see chapter 4, pages 60-90).

**Bach** does not explicitly teach a system for maintaining data wherein usage bits are maintained indicating membership in an active file system and a read-only file system.

**Chutani et al.**, however, teaches a system for maintaining data wherein usage bits are maintained indicating membership in an active file system and a read-only file system (see page 44, fourth full paragraph; see also page 46, seventh paragraph; see also Figures 1 and 2; see also the disclosure that an anode is analogous to an inode in a UNIX system, page 45, first full paragraph).

It would have been obvious to one of ordinary skill in the art at the time of the invention to maintain a record of changes in the storage system, i.e. a fileset clone, since such a feature would be very important to the administrative operators' implementation, since the administrative tools could operate on the clones instead of on the read-write data for their work, greatly reducing the amount of time they require exclusive access to the read-write data (see page 44, fourth full paragraph).

Neither **Bach** nor **Chutani et al.** explicitly teaches a system for maintaining data wherein the storage system is capable of storing multiple usage bits for multiple read-only copies of a file system.

**Nishigaki et al.**, however, teaches a system for maintaining data wherein the storage system is capable of storing multiple read-only copies of a file system (see disclosure of multiple backup

Art Unit: 2177

version page tables and multiple backup version slot maps, analogous to the claimed multiple usage bits and multiple read-only copies of a file system, col. 12, lines 16-46; see also Figure 8).

It would have been obvious to one of ordinary skill in the art at the time of the invention to incorporate the teachings of Nishigaki et al. with those of Bach and Chutani et al., since all three are concerned with the same field of endeavor, that is, managing a file system.

Furthermore, it would have been obvious to one of ordinary skill in the art at the time of the invention to allow the storage of multiple copies of read-only copies of a file system, since this would allow the storage of multiple backup versions (see col. 12, lines 28-32), providing the advantage of allowing the recovery of backups older than merely the last one.

24. Regarding claims 4, 14 and 24, Chutani et al. additionally teaches a memory, system and method of maintaining data wherein one or more bits in each of said plurality of blocks further indicate block reusability (see page 46, first paragraph, citing the storage of information regarding whether a fragment, analogous to the claimed block, is allocated).

It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the references since they both are concerned with the same field of endeavor, that is, managing a file system.

Furthermore, it would have been obvious to one of ordinary skill in the art at the time of the invention to allocate one or more bits to indicate block reusability, since without such an indication, space on the file system could never be recovered, and the storage system would rapidly fill to capacity (see page 46, first paragraph).

25. Claims 5, 15 and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Bach** ("The Design of the UNIX® Operating System") in view of **Thatte** (U.S. Patent 5,008,786).

26. Regarding claims 5 and 15, **Bach** teaches a memory and method of generating a consistency point in a storage system substantially as claimed, comprising the internal representation of files in the UNIX operating system, including the maintenance of inodes and blocks (see chapter 4, pages 60-90).

**Bach** does not explicitly teach a memory and method of generating a consistency point comprising the claimed steps of marking inodes, flushing regular files, flushing special files, flushing blocks of the file system, queuing dirty inodes, and requeuing dirty inodes.

**Thatte**, however, teaches a memory and method of generating a consistency point comprising the steps of:

a) marking a plurality of pages pointing to modified blocks in a file system on said storage system as being in a consistency point (see analogous marking of dirty pages as copy-on-write, col. 16, lines 48-58);

b) flushing regular files to said storage system (see saving of dirty pages, col. 16, lines 34-35);

c) flushing special files to said storage system (see saving of transient root, col. 16, lines 29-30);

d) flushing at least one block of file system information to said storage system (see saving of processor registers and transient root, col. 16, lines 29-30);

Art Unit: 2177

e) queueing dirty inodes after said step of marking and before said step of flushing at least one block of file system information (see disclosure of the trap handler making new copies of pages modified while marked copy-on-write, col. 17, lines 1-8); and

f) requeueing any of said dirty inodes that were not part of said consistency point after said step of flushing at least one block of file system information (see disclosure that none of the newly created dirty pages are cleaned up while copy-on-write operations proceed, col. 17, lines 8-10).

It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the references, since they are both concerned with the same field of endeavor, that is, managing file systems.

Furthermore, it would have been obvious to one of ordinary skill in the art at the time of the invention to incorporate the claimed method of consistency point creation, since copy-on-write techniques allows the resumption of normal operation of the system without any user-perceived pause (see col. 16, lines 48-67).

27. Regarding claim 25, Bach teaches a system for maintaining data in a storage system substantially as claimed, comprising a processor (inherent in a computer system), a storage system (inherent in a file system), and a memory storing information including instructions, the instructions executable by the processor to record a plurality of data about a plurality of blocks of data in the storage system, the instructions comprising the steps of maintaining the internal representation of files in the UNIX operating system, including the maintenance of inodes and blocks (see chapter 4, pages 60-90).

**Bach** does not explicitly teach a system for generating a consistency point comprising the claimed steps of marking inodes, flushing regular files, flushing special files, flushing blocks of the file system, queuing dirty inodes, and requeuing dirty inodes.

**Thatte**, however, teaches a system for generating a consistency point comprising the steps of:

a) marking a plurality of pages pointing to modified blocks in a file system on said storage system as being in a consistency point (see analogous marking of dirty pages as copy-on-write, col. 16, lines 48-58);

b) flushing regular files to said storage system (see saving of dirty pages, col. 16, lines 34-35);

c) flushing special files to said storage system (see saving of transient root, col. 16, lines 29-30);

d) flushing at least one block of file system information to said storage system (see saving of processor registers and transient root, col. 16, lines 29-30);

e) queueing dirty inodes after said step of marking and before said step of flushing at least one block of file system information (see disclosure of the trap handler making new copies of pages modified while marked copy-on-write, col. 17, lines 1-8); and

f) requeuing any of said dirty inodes that were not part of said consistency point after said step of flushing at least one block of file system information (see disclosure that none of the newly created dirty pages are cleaned up while copy-on-write operations proceed, col. 17, lines 8-10).

It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the references, since they are both concerned with the same field of endeavor, that is, managing file systems.

Furthermore, it would have been obvious to one of ordinary skill in the art at the time of the invention to incorporate the claimed method of consistency point creation, since copy-on-write techniques allows the resumption of normal operation of the system without any user-perceived pause (see col. 16, lines 48-67).

28. Claims 7, 10, 17, 20, 27, 30 and 33 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Bach** ("The Design of the UNIX® Operating System") in view of **Chutani et al.** ("The Episode File System") in view of **Thatte** (U.S. Patent 5,008,786).

29. Regarding claims 7 and 17, **Bach** teaches a memory and method of maintaining data in a storage system substantially as claimed, comprising the internal representation of files in the UNIX operating system, including maintaining a root node and inodes for a file system (see Figures 4.6 and 4.10, on pages 69 and 73 respectively), the root node pointing directly or indirectly to the inodes, each inode storing file data, pointing to one or more blocks in the storage system that store file data, or pointing to other inodes (see definition of inodes, pages 61-63, including Figure 4.2; see also discussion of direct and indirect blocks in the file system, pages 68-72, including Figure 4.6).

**Bach** does not explicitly teach a memory and method of maintaining data wherein modified or new data is written to new blocks and inodes in the storage system, with the old blocks and

inodes being maintained, so that a record of changes to the file system is automatically maintained in the storage system.

**Chutani et al.**, however, teaches a memory and method of maintaining data wherein modified or new data is written to new blocks and inodes in the storage system, with the old blocks and inodes being maintained, so that a record of changes to the file system is automatically maintained in the storage system (see page 44, fourth full paragraph; see also page 46, seventh paragraph; see also Figures 1 and 2; see also the disclosure that an anode is analogous to an inode in a UNIX system, page 45, first full paragraph).

It would have been obvious to one of ordinary skill in the art at the time of the invention to maintain a record of changes in the storage system, i.e. a fileset clone, since such a feature would be very important to the administrative operators' implementation, since the administrative tools could operate on the clones instead of on the read-write data for their work, greatly reducing the amount of time they require exclusive access to the read-write data (see page 44, fourth full paragraph).

Neither **Bach** nor **Chutani et al.** explicitly teaches a memory and method of maintaining data wherein new data and inodes affected by the new data are temporarily stored in memory before writing the new data and inodes affected by the new data to the storage system, using a list of dirty inodes to coordinate writing the new data and inodes affected by the new data to new blocks in the storage system.

**Thatte**, however, teaches a memory and method of maintaining data wherein new pages (analogous to the claimed data and inodes) affected by the new data are temporarily stored in memory before writing the new pages affected by the new data to the storage system, using a list of dirty inodes to coordinate writing the new pages affected by the new data to new blocks in the storage system (see discussion of the details of copy-on-write operations, whereby dirty pages are maintained in memory until a new checkpoint is created, when the list of dirty pages is used to coordinate the writing of new pages to disk, col. 16, lines 4-11; see also col. 16, lines 27-37 and 48-58; see also col. 16, line 64 through col. 17, line 17).

It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the three references, since they are all concerned with the same field of endeavor, that is, the maintenance of a file system in a computer.

Furthermore, it would have been obvious to one of ordinary skill in the art at the time of the invention to store new data temporarily in memory and to use the list of inodes to coordinate the writing of said new data to the storage system, since in this way checkpoints can be maintained, which allow the system to recover from hardware failures and software errors, in such a way that the system can be restored to an earlier, valid state to minimize the loss of work (see col. 6, lines 17-26).

30. Regarding claim 27, **Bach** teaches a system for maintaining data in a storage system substantially as claimed, comprising a processor (inherent in a computer system), a storage system (inherent in a file system), and a memory storing information including instructions, the instructions executable by the processor to record a plurality of data about a plurality of blocks of data in the storage system, the instructions comprising the steps of maintaining data in a storage system

Art Unit: 2177

substantially as claimed, comprising the internal representation of files in the UNIX operating system, including maintaining a root node and inodes for a file system (see Figures 4.6 and 4.10, on pages 69 and 73 respectively), the root node pointing directly or indirectly to the inodes, each inode storing file data, pointing to one or more blocks in the storage system that store file data, or pointing to other inodes (see definition of inodes, pages 61-63, including Figure 4.2; see also discussion of direct and indirect blocks in the file system, pages 68-72, including Figure 4.6).

**Bach** does not explicitly teach a system for maintaining data wherein modified or new data is written to new blocks and inodes in the storage system, with the old blocks and inodes being maintained, so that a record of changes to the file system is automatically maintained in the storage system.

**Chutani et al.**, however, teaches a system for maintaining data wherein modified or new data is written to new blocks and inodes in the storage system, with the old blocks and inodes being maintained, so that a record of changes to the file system is automatically maintained in the storage system (see page 44, fourth full paragraph; see also page 46, seventh paragraph; see also Figures 1 and 2; see also the disclosure that an anode is analogous to an inode in a UNIX system, page 45, first full paragraph).

It would have been obvious to one of ordinary skill in the art at the time of the invention to maintain a record of changes in the storage system, i.e. a fileset clone, since such a feature would be very important to the administrative operators' implementation, since the administrative tools could

Art Unit: 2177

operate on the clones instead of on the read-write data for their work, greatly reducing the amount of time they require exclusive access to the read-write data (see page 44, fourth full paragraph).

Neither **Bach** nor **Chutani et al.** explicitly teaches a system for maintaining data wherein new data and inodes affected by the new data are temporarily stored in memory before writing the new data and inodes affected by the new data to the storage system, using a list of dirty inodes to coordinate writing the new data and inodes affected by the new data to new blocks in the storage system.

**Thatte**, however, teaches a system for maintaining data wherein new pages (analogous to the claimed data and inodes) affected by the new data are temporarily stored in memory before writing the new pages affected by the new data to the storage system, using a list of dirty inodes to coordinate writing the new pages affected by the new data to new blocks in the storage system (see discussion of the details of copy-on-write operations, whereby dirty pages are maintained in memory until a new checkpoint is created, when the list of dirty pages is used to coordinate the writing of new pages to disk, col. 16, lines 4-11; see also col. 16, lines 27-37 and 48-58; see also col. 16, line 64 through col. 17, line 17).

It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the three references, since they are all concerned with the same field of endeavor, that is, the maintenance of a file system in a computer.

Furthermore, it would have been obvious to one of ordinary skill in the art at the time of the invention to store new data temporarily in memory and to use the list of inodes to coordinate the

writing of said new data to the storage system, since in this way checkpoints can be maintained, which allow the system to recover from hardware failures and software errors, in such a way that the system can be restored to an earlier, valid state to minimize the loss of work (see col. 6, lines 17-26).

31. Regarding claim 33, **Bach** teaches a system for maintaining data in a storage means substantially as claimed, comprising means for maintaining the internal representation of files in the UNIX operating system, including means for maintaining a root node and inodes for a file system (see Figures 4.6 and 4.10, on pages 69 and 73 respectively), the root node pointing directly or indirectly to the inodes, each inode storing file data, pointing to one or more blocks in the storage system that store file data, or pointing to other inodes (see definition of inodes, pages 61-63, including Figure 4.2; see also discussion of direct and indirect blocks in the file system, pages 68-72, including Figure 4.6).

**Bach** does not explicitly teach a system for maintaining data wherein modified or new data is written to new blocks and inodes in the storage system, with the old blocks and inodes being maintained, so that a record of changes to the file system is automatically maintained in the storage system.

**Chutani et al.**, however, teaches a system of maintaining data wherein modified or new data is written to new blocks and inodes in the storage system, with the old blocks and inodes being maintained, so that a record of changes to the file system is automatically maintained in the storage system (see page 44, fourth full paragraph; see also page 46, seventh paragraph; see also Figures 1

Art Unit: 2177

and 2; see also the disclosure that an anode is analogous to an inode in a UNIX system, page 45, first full paragraph).

It would have been obvious to one of ordinary skill in the art at the time of the invention to maintain a record of changes in the storage system, i.e. a fileset clone, since such a feature would be very important to the administrative operators' implementation, since the administrative tools could operate on the clones instead of on the read-write data for their work, greatly reducing the amount of time they require exclusive access to the read-write data (see page 44, fourth full paragraph).

Neither **Bach** nor **Chutani et al.** explicitly teaches a system for maintaining data wherein new data and inodes affected by the new data are temporarily stored in memory before writing the new data and inodes affected by the new data to the storage system, using a list of dirty inodes to coordinate writing the new data and inodes affected by the new data to new blocks in the storage system.

**Thatte**, however, teaches a system for maintaining data including means wherein new pages (analogous to the claimed data and inodes) affected by the new data are temporarily stored in memory before writing the new pages affected by the new data to the storage system, using a list of dirty inodes to coordinate writing the new pages affected by the new data to new blocks in the storage system (see discussion of the details of copy-on-write operations, whereby dirty pages are maintained in memory until a new checkpoint is created, when the list of dirty pages is used to coordinate the writing of new pages to disk, col. 16, lines 4-11; see also col. 16, lines 27-37 and 48-58; see also col. 16, line 64 through col. 17, line 17).

It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the three references, since they are all concerned with the same field of endeavor, that is, the maintenance of a file system in a computer.

Furthermore, it would have been obvious to one of ordinary skill in the art at the time of the invention to store new data temporarily in memory and to use the list of inodes to coordinate the writing of said new data to the storage system, since in this way checkpoints can be maintained, which allow the system to recover from hardware failures and software errors, in such a way that the system can be restored to an earlier, valid state to minimize the loss of work (see col. 6, lines 17-26).

32. Regarding claims 10, 20 and 30, **Chutani et al.** additionally teaches a memory, system and method further comprising the step of creating a clone (analogous to the claimed snapshot) of the system by copying the root node (see page 46, seventh paragraph, which teaches that the creation of a clone is performed by cloning *each* of the anodes, which would include the root anode).

It would have been obvious to one of ordinary skill in the art at the time of the invention to create a snapshot as claimed, since creation of the snapshot (clone) is important to the administrative operation of the system (see page 44, fourth full paragraph, et seq.), providing the advantage of allowing checkpoints to be created which minimize the loss of work in case of a system failure.

33. Claims 11, 21 and 31 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Bach** ("The Design of the UNIX® Operating System") in view of **Chutani et al.** ("The Episode File

Art Unit: 2177

System") in view of **Thatte** (U.S. Patent 5,008,786) as applied to claims 7, 10, 17, 20, 27, 30 and 33 above, and further in view of **Raz** (U.S. Patent 5,701,480).

34. Regarding claims 11, 21 and 31, **Bach, Chutani et al.** and **Thatte** teach a memory, system and method of maintaining data in a storage system substantially as claimed.

None of **Bach, Chutani et al.** nor **Thatte** explicitly teaches a memory, system and method of maintaining data in a storage system wherein the block map indicates membership of blocks in one or more snapshots.

However, **Raz** teaches a memory, system and method of maintaining data in a storage system wherein the block map indicates membership of blocks in one or more snapshots (see discussion of the snapshot pages, col. 61, line 7 through col. 62, line 2, and particularly col. 61, lines 39-46; see also Figures 23 and 26).

It would have been obvious to one of ordinary skill in the art at the time of the invention to maintain information on the membership of blocks in one or more snapshots, since this provides a way for the system to identify the snapshot records necessary to carry out operations for which the live version is not the proper version (see col. 60, lines 31-51).

35. Claims 12, 22 and 32 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Bach** ("The Design of the UNIX® Operating System") in view of **Chutani et al.** ("The Episode File

Art Unit: 2177

System") in view of **Thatte** (U.S. Patent 5,008,786) as applied to claims 7, 10, 17, 20, 27, 30 and 33 above, and further in view of Applicants' **Admitted Prior Art**.

36. Regarding claims 12, 22 and 32, **Bach, Chutani et al.** and **Thatte** teach a memory, system and method of maintaining data in a storage system substantially as claimed.

None of **Bach, Chutani et al.** nor **Thatte** explicitly teaches a memory, system and method of maintaining data in a storage system further comprising the step of deleting a snapshot from the storage system wherein blocks that are only part of the deleted snapshot are released for re-use by the storage system.

However, Applicants' **Admitted Prior Art** teaches that in the Episode File System, the fileset clone, analogous to the claimed snapshot, is deleted when the backup has completed (see specification at page 7, lines 9-22, particularly lines 19-20).

It would have been obvious to one of ordinary skill in the art at the time of the invention to delete snapshots from the storage system and release those freed blocks which are only part of the deleted snapshot to the storage system for reuse, since without the capacity to delete snapshots and release the blocks for reuse, any memory used for snapshots would be lost thereafter, and the system would eventually run out of resources. The claimed limitation constitutes garbage collection, a practice that was well known in the art at the time of the invention, and one whose use would have been obvious to an ordinary artisan.

*Allowable Subject Matter*

37. Claims 6, 16 and 26 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

38. The following is a statement of reasons for the indication of allowable subject matter:

The present invention is directed to a method and system for maintaining a file system wherein membership in multiple concurrently existing file systems is possible for each block, and wherein consistency points are created in part by flushing special files to the storage system.

**Bach** teaches a system for maintaining data in a storage system comprising means for maintaining the internal representation of files in the UNIX operating system, including means for maintaining a root node and inodes for a file system (see Figures 4.6 and 4.10, on pages 69 and 73 respectively), the root node pointing directly or indirectly to the inodes, each inode storing file data, pointing to one or more blocks in the storage system that store file data, or pointing to other inodes (see definition of inodes, pages 61-63, including Figure 4.2; see also discussion of direct and indirect blocks in the file system, pages 68-72, including Figure 4.6). However, **Bach** does not explicitly teach a system for maintaining data wherein consistency points are generated, as recited in dependent claims 6, 16 and 26.

**Thatte** teaches a system for generating a consistency point, including the step of flushing special files to said storage system (see saving of transient root, col. 16, lines 29-30). However, **Thatte** fails to teach the details of the flushing step as recited in dependent claims 6, 16 and 26.

These features, together with the other limitations of the cited dependent claims are novel and non-obvious over the prior art of record.

### *Response to Arguments*

39. Applicant's arguments with respect to claims 1-33 have been considered but are moot in view of the new ground(s) of rejection.

### *Conclusion*

40. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

**Feldman et al.** ("IGOR: A System for Program Debugging via Reversible Execution") teaches details of the practice of checkpointing.

**Li et al.** ("Real-Time, Concurrent Checkpoint for Parallel Programs") teaches details of different checkpointing algorithms.

**Plank et al.** ("Libckpt: Transparent Checkpointing under Unix") teaches the details of different checkpointing algorithms.

Art Unit: 2177

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Luke S. Wassum whose telephone number is 703-305-5706. The examiner can normally be reached on Monday-Friday 8:30-5:30, alternate Fridays off.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John E. Breene can be reached on 703-305-9790. The fax phone numbers for the organization where this application or proceeding is assigned are 703-746-7239 for regular communications and 703-746-7238 for After Final communications.

In addition, INFORMAL or DRAFT communications may be faxed directly to the examiner at 703-746-5658.

Customer Service for Tech Center 2100 can be reached during regular business hours at (703) 306-5631, or fax (703) 746-7240.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is 703-305-3900.



Luke S. Wassum  
Art Unit 2177

lsw  
May 14, 2003